ISSN 2186-5647



2-42

# コンカレントシステムに対する抽象化タイムスタンプを用いたSPINモデル検査手法

阪大・情報科学（院）　○中野 伸哉
阪大・情報科学　土屋 達弘

## 1. はじめに

並行アルゴリズムや分散アルゴリズムの設計は難しく，捉えにくい設計誤りが含まれることが多い．このようなアルゴリズムの検証手法として，状態空間の探索に基づいた技術であるモデル検査が有用であることが知られている．しかしながら，アルゴリズムが上界のないタイムスタンプを用いている場合，モデル検査の適用は困難となる．これは，状態空間が無限となってしまうからである．本論文では，非有界なタイムスタンプに対する抽象化を提案し，無限の状態空間を有するアルゴリズムを，その過大近似であるような有限状態の抽象化アルゴリズムに変換する手法を提案する．非有界なタイムスタンプを有するアルゴリズムであるLamportのベーカリーアルゴリズム[1]とAndrewsのチケットアルゴリズム[2]を対象に，提案手法を用いてモデル検査を行うことで，提案手法の妥当性を示す．

## 2. 状態遷移システム

コンカレントシステムのアルゴリズムは複雑であるために，アルゴリズムに対する検証，証明が有用である．検証手法の一つとしてモデル検査が挙げられる．この方法では対象のアルゴリズムをモデル化し，モデル検査器を用いて検証を行う．検証されるアルゴリズムは状態遷移システムによって表すことができ，それに則して検証される．状態遷移システムは3個のタプル$(\boldsymbol{S}, \boldsymbol{T}, s_{init})$から成る．

- $\boldsymbol{S}$: 状態集合であり，$\boldsymbol{S} \equiv D_1 \times \ldots \times D_F \times \boldsymbol{N}_0^N$と表すことができる．ここで，$\mathrm{D_i}$は$i$番目の有限状態変数の定義域，$F$は有限状態変数の個数であり，$\boldsymbol{N}_0$は非負整数の集合である．
- $\boldsymbol{T} \subseteq \boldsymbol{S} \times \boldsymbol{S}$: 遷移集合である．遷移関係を$(s, s')$の組で表し，ガード条件が真のとき，動作として状態$s$から次状態$s'$へ遷移することを示す．
- $s_{init} \in \boldsymbol{S}$: 初期状態である．

いくつかのアルゴリズムにはタイムスタンプと呼ばれる非有界変数が用いられているので，状態集合を生成する際に，状態爆発を起こし，検証を行うことができない．よって，非有界変数を有界変数とみなして扱い検証を行う方法を提案する．

## 3. 抽象化タイムスタンプ

提案する抽象化手法の基本的な発想は，全タイムスタンプが$N$個あったとき，タイムスタンプを$[0, N]$の範囲に写像することである．タイムスタンプは他のタイムスタンプとの大小関係が重要であるため，大小関係を保証しながら抽象化を行わなければならない．また，$N+1$個の整数値を用いる理由は，タイムスタンプが0か0でないかを区別するために，0を用いるためである．この写像を$h$とする．

$$h: \boldsymbol{N}_0^N \rightarrow \{0, \ldots, N\}^N$$

ただし，$\boldsymbol{N}_0$は非負整数の集合である．さらに，$h(\cdot)$は$i$番目の要素とする写像h(·)とすると，写像$h(\cdot)$は以下のように定義することができる．

$$h_i(ts_1, \ldots, ts_N) = \begin{cases} 0 & ts_i = 0 \\ j & ts_i \neq 0 \text{ かつ} ts_i \text{は} j \text{番目} \\ & \text{に 0 を除いて小さい} \\ & \{ts_i (1 \leq i \leq N) | ts_i \neq 0\} \end{cases}$$

次に写像 $h(\cdot)$ の具体例を示す．

- $h(15, 0, 15) = (1, 0, 1)$

  $ts_2 = 0$ なので $h_2(15, 0, 15)$ は 0 のままである．
- $h(10, 14, 8) = (2, 3, 1)$

$ts_i$が 0 を含んでいないので，一番小さい$ts_3 = 8$から順番に 1 から値が割り当てられる．

上記の$h(\cdot)$に従って，無限に発散するタイムスタンプを持たない新たなアルゴリズムに元のアルゴリズムを書き換える．さらに，タイムスタンプの代入操作を以下の 3 操作に制限する．

| | |
|---|---|
| i. Reset) | $ts_i := 0$ |
| ii. Set) | $ts_i := ts_j \quad (i \neq j)$ |
| iii. Set and increment) | $ts_i := ts_j + 1$ |

## 4. 検証項目

本研究が想定している検査方法は，満たすべき性質を線形時相論理式(LTL)で表現し，この性質の否定を表す never オートマトンとモデルを表すオートマトンとの積をとることによって合成を行い，モデル検査を行

---

Model checking using abstract timestamps for concurrent systems

Shinya NAKANO and Tatsuhiro TSUCHIYA

う．アルゴリズムによって保証されるべき以下の 2 種類の性質を検査項目として設定する．

i. ベーカリーアルゴリズム

- 相互排除: $\square(num_{cs} \leq 1)$
- プロセスの進行性:
  $\square(choosing_{n-1} = true \Rightarrow\diamond\ in_{cs_{n-1}} = true)$

ii. チケットアルゴリズム

- 相互排除: $\square(num_{cs} \leq 1)$
- プロセスの進行性:
  $\square(ticket_0 \neq 0 \Rightarrow\diamond\ in_{cs_0} = true)$

## 5. 実験結果

3 節で述べた抽象化手法を実際のアルゴリズムに適用し，モデル検査を行う．本論文ではベーカリーアルゴリズムとチケットアルゴリズムを対象にした．これらのアルゴリズムはコンカレントシステムにおける相互排除を実現するための古典的なアルゴリズムである．プロセスの総数を$n$とし，プロセス番号を$0,\ldots,n-1$の範囲で割り当てる．

実験環境としては，有限状態モデル検査器として有名な SPIN[3]を使用し, Linux (CentOS)の環境を用いた．CPU は Xeon E5-2665，DRAM は 128GB のメモリをそれぞれ使用した．本論文で提案した抽象化手法を適用したモデルに対して, Promela 言語を用いて実装し実験を行った. 実験結果を表 1 に示す. 実験結果としては，ベーカリーアルゴリズムに関しては検査項目に違反は見つからず，このアルゴリズムは検査項目については正しいということが示された．しかしながら，チケットアルゴリズムの相互排除に関しては違反が見つからなかったが，プロセスの進行性に関しては違反が検出された．これはタイムスタンプの抽象化に伴う，モデルの過大近似によるものである．実際のアルゴリズムでは起こりえない動作が，タイムスタンプの Set and increment 操作の過大近似により，生じてしまったことが原因である．これに伴い，クリティカルセクションに入る条件がすべてのプロセスで満足できず，デッドロックが生じてしまうことがわかった．

**表 1　実験結果**

**(1.a)　ベーカリーアルゴリズムの相互排除**

| | n = 2 | n = 3 | n = 4 |
|---|---|---|---|
| #States | 17840 | 1.64E+06 | 3.79E+08 |
| #Transitions | 34555 | 4.70E+06 | 1.44E+09 |
| Elapsed time[sec] | 0.18 | 45.5 | 2.23E+04 |

**(1.b)　ベーカリーアルゴリズムのプロセスの進行性**

| | n = 2 | n = 3 | n = 4 |
|---|---|---|---|
| #States | 17840 | 3.20E+06 | NA |
| #Transitions | 34555 | 1.81E+06 | NA |
| Elapsed time[sec] | 0.61 | 152 | NA |

**(2.a)　チケットアルゴリズムの相互排除**

| | n = 2 | n = 3 | n = 4 |
|---|---|---|---|
| #States | 8 | 135 | 1492 |
| #Transitions | 10 | 330 | 2215 |
| Elapsed time[sec] | 0 | 0.052 | 0.437 |

**(2.b)　チケットアルゴリズムのプロセスの進行性**

| | n = 2 | n = 3 | n = 4 |
|---|---|---|---|
| #States | 34 | 142 | 290 |
| #Transitions | 65 | 359 | 690 |
| Elapsed time[sec] | 0.005 | 0.037 | 0.089 |

## 6. まとめ

本研究では非有界のタイムスタンプが使われているコンカレントシステムに対して，モデル検査の手法を提案した．基本的な発想としては，タイムスタンプは大小関係を調べるために使われているので，有限の状態に変換を行うというものである．

有限状態に抽象化を行うことができれば，状態遷移システムの状態空間もまた有限になることから，有限モデル検査器を用いて，これらのシステムがモデル検査を行うことができる．本研究では有限状態モデル検査器SPINを用いて，ベーカリーアルゴリズムとチケットアルゴリズムを対象に実験を行った．チケットアルゴリズムに関しては過大近似による制約違反が検出され，プロセスの進行性に関しては検査することができなかった．今後の課題として，過大近似による制約違反の誤検出を防ぐため，抽象化手法の改良を検討する．

また，コンセンサスアルゴリズムのひとつである RAFT[4]のリーダーエレクションアルゴリズムに対して，本手法を適用する．

### 参考文献


[1] L. Lamport, “A new solution of Dijkstra’s concurrent programming problem,” Commun. ACM, vol.17, no.8, pp.453-455, Aug. 1974. http://doi.acm.org/10.1145/361082.361093

[2] G. Andrews. Concurrent programming: principles and practice. Benjamin/Cummings Publishing Company, 1991.

[3] G.J. Holzmann, The SPIN Model Checker, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2004.

[4] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp.305–319, 2014.